UM TAGUM COLLEGE

DEPARTMENT OF ENGINEERING EDUCATION

COMPUTER ENGINEERING PROGRAM

VERILOG HDL SYNTAX

AND SEMANTICS

DRILL 2

NAME:

STUDENT NUMBER:

TERMINAL NUMBER:

DATE OF PERFORMANCE:

DATE OF SUBMISSION:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROFESSOR

I. DISCUSSION

Besides a module, the following are the list of major elements that make up a Verilog HDL program.

**A. Value Sets**

There are three types of constants in HDL:

* 0 – logic value false
* 1 – logic value true
* x – unknown
* z – high impedance

**B. Data Types**

Verilog has two data types: a *net* type representing a physical connection between structural elements, and a *register* type that represents an abstract storage element.

**C. Constants**

The three types of constants in Verilog HDL are the following:

1. Integers

An integer number can be written in either simple decimal (with or without unary operator) or base format (in the form [size]’base value).

1. Real

A real number can be specified in either decimal or scientific notation.

1. String

A string is a sequence of characters within double quotes.

**D. Operators**

Operators are classified into the following:

1. Arithmetic (+, -, \*, /, %)
2. Relational (>, >=, <, <=)
3. Equality (==, !=)
4. Logical (&&, ||, !)
5. Bitwise (~, &, |, ^, ~^ or ^~)
6. Reduction (&, ~&, |, ~|, ^, ~^)
7. Shift (<<, >>)
8. Conditional (cond?expr1:expr2)

**E. Vectors**

Some gate-level modelling use identifiers having multiple bit widths, called vectors. The syntax specifying a vector includes within square brackets two numbers separated with a colon.

output [0:7] A;

wire [5:0] B;

The individual and contiguous bits are specified within square brackets, so B[2] specifies bit 2 of B, and B[2:0] specifies the three least significant bits of B.

**F. Parameters**

A parameter is a constant used to specify delays and widths of variables.

*parameter Sum=2’b00, Diff=2’b11;*

**G. Identifiers**

Identifiers are names used to give an object, such as a register or a module, a name so that it can be referenced from other places in a description.

* Identifiers must begin with an alphabetic character or the underscore character ( a-z A-Z)
* Identifiers may contain alphabetic characters, numeric characters, the underscore, and the dollar sign ( a-z A-Z 0-9 \_ $ )
* Identifiers can be up to 1024 characters long.

**H. Directives**

Compiler directives are identifiers that start with the ` (backquote) character.

* **`define** and **`undef**

The **`**define directive is used to define a text macro while **`**undef cancels a macro definition.

* **`ifdef, `else** and **`endif**

These directives are used for conditional compilation.

* **`default\_nettype**

This directive is used to specify the net type for implicit declarations.

* **`include**

This directive directs the compiler to read the contents of a file in-line.

* **`resetall**

This directive resets all active compiler directives to their default values.

* **`timescale**

This directive defines the simulation time units and precision.

* **`unconnected\_drive** and **`nounconnected\_drive**

The `unconnected\_drive causes unconnected inputs of modules to pull up or down while the nounconnected\_drive resumes the default, which is to make unconnected inputs float with a value of Z.

* **`celldefine** and **`endcelldefine**

This directive is used before and after a module to tag it as a library cell and are used by certain PLI routines for applications such as delay.

**I. Keywords**

These are reserved words that cannot be used as identifiers. See Appendix A for the list of all Verilog HDL keywords.

II. Drill Exercises

A. Drill2\_1

Code the program below; save it as file drill2\_1.

*//This Verilog test bench illustrates the different Verilog constants*

*module drill2\_1;*

*reg [7:0] a, b, c, d, e, f, g, h, i;*

*reg [9:0] j;*

*initial begin*

*a=5’O37;*

*b=2’D4;*

*c=7’hx;*

*d=8‘ h AF;*

*e=10’b01;*

*f=(2+3)’d10;*

*g=32;*

*h=4’d-5;*

*i=”Testing”;*

*j=3.6E2;*

*$write(" %o %b %b %h %b",a,b,c,d,e);*

*$write(" %d %d %d %d",f,g,h,i,j);*

*end*

*endmodule*

Run the program in your command prompt. You may notice that there will be list of errors that will be shown, with the format similar with the following

*constants.vl:6: warning: Numeric constant truncated to 2 bits.*

*constants.vl:12: error: unmatched character (‘)*

Resolve the issues by doing the following modifications:

* Remove register variable i.
* Remove declaration of string; display it directly within the $write() function
* Change (2+3)’d10 to 2+3’d10, 2’D4 to 3’D4, 8’ h AF to 8 ‘h AF
* Assign h with -5 (use simple decimal instead of base format)

Run the program again, and observe the output.

B. Drill2\_2

Code the following program and save it under the filename drill2\_2.

*//This Verilog test bench illustrates the different Verilog operators*

*module drill2\_2;*

*reg [3:0]Ctr, Xnr, Fdr;*

*initial begin*

*Ctr=4’d0; Fdr=4’d11;*

*#5 Xnr = (Ctr !=0) ? (Ctr \*+ 1) : ^Fdr;*

*#10 $display(Xnr);*

*#15 Xnr=-9%2; Fdr=52<8’hFF;*

*#20 $display(Xnr,” “,Fdr);*

*#25 Ctr=Xnr&&Fdr; Fdr=Xnr&Fdr;*

*#30 $display(Ctr,Fdr);*

*#35 Ctr=Ctr<<2; Fdr=Xnr>>1;*

*#40 $display(Ctr, Fdr);*

*#100 $finish;*

*end*

*endmodule*

Analyze the output of the operations involved in the test bench provided above by simulating it in your command prompt.

C. Drill2\_3

*`timescale 10 ns / 100 ps*

*/\*The time unit is set to 10 ns. During the simulation all delay values will be multiplied by 10 ns, and all delays will be rounded with 100 ps precision.\*/*

*`define SIZE 8*

*`define STOP $finish*

*`define REGME reg [8\*31:0]*

*`ifdef TEST\_1*

*`include "drill1\_1.vl"*

*`else*

*`include "drill1\_2.vl"*

*`endif*

*module testbench;*

*`REGME regVar;*

*reg a, b,borrowIn;*

*wire diff, borrowOut;*

*full\_subtract fs(diff, borrowOut, a, b, borrowIn);*

*initial begin*

*a=1'b1; b=1'b1; borrowIn=1'b0;*

*end*

*initial begin*

*#10 a=1'b1;*

*#10 a=1'b0; b=1'b1;*

*#10 a=1'b1; b=1'b0;*

*#10 borrowIn=1'b1;*

*end*

*initial begin*

*$display(" a b borrowIn difference borrowOut time");*

*$monitor(“%b %b %b %b %b %d“, a, b, borrowIn, diff, borrowOut, $time);*

*#10 `STOP;*

*end*

*endmodule*

III. Programming Exercise

1. Create a test bench that will actually perform the requirements specified below. Save as exercise2\_1.vl

* *Display a sized hexadecimal value assigned in a variable.*
* *Display a real number defined within your program as <mantissa> E <exponent>*
* *Display a backslash (\), percent(%), a new tab, and a double quote(“) character within a string.*
* *Display a hardcoded string “Hello” stored in a 20Bytes register. Make sure to avoid displaying leading spaces or zeros.*
* *Illustrate use of conditional operators*

1. Use a file from your previous drills (drill1\_1.vl or drill1\_2.vl) and include it inside your own file. Define a new time scale of 100 ns time precision and 1s time units. Allow only the new test bench to work instead of the ones saved in drill1\_1.vl or drill1\_2.vl. Save as exercise2\_2.vl.
2. Define a module that will compute for the volume of a cylinder using the formula . Define the value of pi using `define. Use the module to display the volume of a hardcoded value in the form *Ans*r*Rem*. Avoid having leading zeros or spaces in your answers. Save as exercise2\_3.vl

IV. Review Questions

1. What is the difference between vectors X[5:0] and X[0:5]?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. How are negative numbers represented? How does Verilog treat these numbers?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What happens in integer numbers if the size is greater than the value? What if the size is less than the value?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. When are white spaces significant in your programs?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Evaluate the following:

A=8’b10101010 B=8’b01101110 C=8’b11101011

* 1. B&C
  2. |A
  3. ~^C
  4. A||B
  5. A>C
  6. B<=A
  7. C<<2
  8. B\*2
  9. C%A
  10. A-B

1. When is *x (unknown or don’t care)* assigned during simulation, and when does high impedance *z* occurs?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_